Circuit Simulation Code Generation by Computer Algebra(Software Science and Engineering)

AUTHOR(S):

Loe, K.G.F.; Ohsawa, N.; Goto, E.

数理解析研究所講究録
第547巻 1985年 287-302

Circuit Simulation Code Generation by Computer Algebra

K. F. Loe[*1,*2,*3] (盧 加福)
N. Ohsawa[*1] (大澤範高)
E. Goto[*1,*2] (後藤英一)

ABSTRACT

A simulation program generator, which generates circuitry code for circuit simulation with input of circuit specification, is developed based on computer algebra algorithms and Hamiltonian formalism. The generator is easy to use and extensible to include new logic function.

1. Introduction

In the process of developing a computer hardware, circuit simulation is essential, since it provide the hardware designer with many useful information prior to the actual hardware design. Basically there are two possible ways to perform circuit simulations with software programming. The most primitive method is for the user to code his own program based on some first principles of circuit design. However, this method is not only time consuming, but also

*1 University of Tokyo, Faculty of Science, Dept. of Information Science,
*2 The Institute of Physical and Chemical Research, Information Science Lab.,
*3 National University of Singapore, Dept. of Computer Science.

much error prone for a complicated system. The other method is to use some software packages of circuit simulation available in the market. While it may be convenience to do so for a conventional circuit design, it would be difficult to modify the program to suit some particular applications which are not available in the simulator packages. For example, if we are to apply the conventional circuit simulator to the problems of Josephson junction circuitry, probably many changes may have to be made to the circuit simulator since in the superconductivity domain different principles of circuit operation are applied.

In the following we propose a third method which is to design an circuit simulation code generator (CSCG) for circuit simulation. CSCG is not only easy to use, but also extensible to allow the incorporation of new logic functions into it as will be detailed in the following. One of the limitation of this approach is that the system must be able to formulate using Hamiltonian formalism. However the approach is particular superior in the application to the DCFP[1] , which is a kind of Josephson junction device intended for building very high speed future computer system. Since writing down the Hamiltonian or Lagrangian for a complicated circuit of DCFP would be easier than using the conventional circuit analysis method, thus this tool is particularly suitable for DCFP logic design.

In fact, the generator is built up from some modularity of sub-Hamiltonians and the total Hamiltonian of the system need to be simulated is the sum of many of these sub-Hamiltonians. Owing to the modularity of the Hamiltonian, the system is extensible, therefore an innovative user can design his own new logic component. To incorporate a new logic component into the system, the user only has to write the coding of the sub-Hamiltonian which corresponds to the new logic component he intended to design . In the other

hand, for a user to use the generator would be very much easier, he only has to input the specifications of the circuit and FORTRAN code can be automatically generated from the circuit specifications. Hamiltonian and the computer algebra based on REDUCE(3.0)[2] are essential in our design of the generator. In the following the principles of Lagrangian and Hamiltonian formalism would be illustrated using a simple example and the applications of REDUCE(3.0) system will be given to illustrate the underlying principle.

2. Formalism of Lagrangian and Hamiltonian

A circuit which consists of superconductive inductances and Josephson junctions with flux inputs and outputs can be easily specified by writing down the potential energy U, the kinetic energy K and the dissipating function D. the Lagrangian is given by the difference of the kinetic energy and the potential energy, the Hamiltonian is given by the sum of kinetic energy, which must be written in term of canonical momentum p, and the potential energy. The Lagrangian and the Hamiltonian of the system are as follow,

$$L = K - U \quad \text{and} \quad H = K_p + U \tag{1}$$

where $K_p = K_p(p_1, p_2, \ldots, p_n)$ and p_k is the canonical momentum related to the Lagrangian formalism by,

$$p_k = \frac{\partial L}{\partial \dot{x}_k} \tag{2}$$

Given the Lagrangian we can derive a set of simultaneous linear equations from (2). Solving these equations we get, $\dot{x}_k = f_k(p_1, p_2, \ldots, p_n)$ and substitute this into K and $(\frac{\partial D}{\partial \dot{x}_k})$ we obtain K_p and $(\frac{\partial D}{\partial \dot{x}_k})_p$ where $k=1, \ldots, n$.

In some systems equation (2) may have extraneous variables which are linearly dependent on other variables, thus posing singularity problem to the

method of linear equations solving. REDUCE(3.0) provides a linear equations solving facility which will return a message for the set of equations having singularity problem.

The Lagrangian form of equations of motion is given by,

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}_k}\right) - \frac{\partial L}{\partial x_k} = -\frac{\partial D}{\partial \dot{x}_k} \tag{3}$$

The Hamiltonian form of equations of motion is given by,

$$\frac{dx_k}{dt} = \frac{\partial H}{\partial p_k} \tag{4}$$

$$\frac{dp_k}{dt} = -\frac{\partial H}{\partial x_k} - \left(\frac{\partial D}{\partial \dot{x}_k}\right)_p \tag{5}$$

3. A Simple Example of Lagrangian and Hamiltonian Formalism

We will derive the Lagrangian and the Hamiltonian equations of motion for a simple example to show the overview of this method.

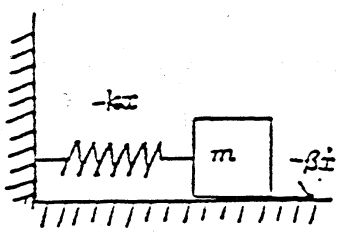


Fig.1

Fig.1 shows a harmonic oscillator with x denotes the coordinate and p denotes the canonical momentum. Also mass of the particle is m, the restitutional force is $-kx$, the friction is $-\beta\dot{x}$. The potential energy U, kinetic energy K and dissipating function D are respectively given as follow,

$$U = \frac{k}{2}x^2 \tag{6}$$

$$K = \frac{m}{2}\dot{x}^2 \tag{7}$$

$$D = \frac{\beta}{2}\dot{x}^2 \tag{8}$$

the Lagrangian L is

$$\begin{aligned} L &= K - U \\ &= \frac{m}{2}\dot{x}^2 - \frac{k}{2}x^2 \end{aligned} \tag{9}$$

and the canonical momentum p is,

$$\begin{aligned} p &= \frac{\partial L}{\partial \dot{x}} \\ &= m\dot{x} \end{aligned} \tag{10}$$

then K_p and $(\frac{\partial D}{\partial \dot{x}})_p$ are obtainable as follow,

$$K_p = \frac{1}{2m}p^2 \tag{11}$$

$$(\frac{\partial D}{\partial \dot{x}})_p = \frac{\beta}{m}p \tag{12}$$

Therefore,

$$\begin{aligned} H &= K_p + U \\ &= \frac{1}{2m}p^2 + \frac{k}{2}x^2 \end{aligned} \tag{13}$$

According to (4) and (5) the equations of motion are

$$\frac{dx}{dt} = \frac{1}{m}p \tag{14}$$

$$\frac{dp}{dt} = \frac{k}{2}x - \frac{\beta}{m}p \tag{15}$$

4. Computer Algebra Algorithms for Runge-Kutta method

In this section, we will show how to use computer algebra based on REDUCE(3.0) to write the algorithm for Runge-Kutta method to solve the Hamiltonian of the abovementioned harmonic motion system. In REDUCE(3.0) the Hamiltonian equations of motion can be written as follows:

$$\dot{x} := DF(H, p) \tag{16}$$

$$\dot{p} := -DF(\mathrm{H}, x) - SUB(\dot{x} = \frac{p}{m}, DF(\mathrm{D}, \dot{x})) \tag{17}$$

where DF() is the differentiate operator of the REDUCE(3.0). If we are to find the algebra expressions for Runge-Kutta method we need to define two algebra parameters HH and TT which are the step of time interval for numerical analysis and the total time of system evolution respectively. The program of this system writing in REDUCE(3.0) is given in List 1. With reference to List 1 we have RUNGEKUTTA(...) which is a procedure for Runge-Kutta algorithms, and the SUB(...) in this procedure is a REDUCE(3.0) system function which is to substitute the algebra value of all the arguments of SUB(...) to the last argument of SUB(...) which are given by either (16) or (17). Using algebra program we can generate the FORTRAN code captured in List 2.

5. Applications of Hamiltonian Formalism to Josephson Junction Circuitry

Here and in the subsequent illustration, X_i and x_i will be used to denote flux and phase at some points i of the circuit respectively, Φ_0 is the unit quantum flux of superconductivity, I_m is the maximum supercurrent of a Josephson junction, and define

$$x_i = 2\pi \frac{X_i}{\Phi_0} \tag{18}$$

$$I_m = \frac{\Phi_0}{2\pi \mathrm{L}_j} \tag{19}$$

$$E_j = \frac{{\Phi_0}^2}{4\pi^2 \mathrm{L}_j} \tag{20}$$

$$\mathrm{L}_i = A_i \mathrm{L}_j \tag{21}$$

$$\tau = \sqrt{\mathrm{C}\mathrm{L}_j} \tag{22}$$

To illustrate the application of the above formalisms to the Josephson junction circuit, we will derive the Hamiltonian equations of motion for Fig. 2

System.

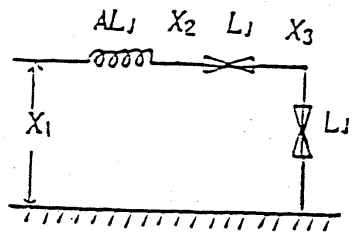


Fig.2

For reason of simplicity in our illustration at here and the next example we assume D=0. Thus the potential energy and kinetic energy of the system are,

$$U = \frac{(X_1-X_2)^2}{2L} - \frac{\Phi_0 I_m}{2\pi}\left[\cos\left(\frac{2\pi(X_2-X_3)}{\Phi_0}\right) + \cos\left(\frac{2\pi X_3}{\Phi_0}\right)\right]$$

$$= E_j\left[\frac{(x_1-x_2)^2}{A} - \cos(x_2-x_3) - \cos x_3\right] \tag{23}$$

$$K = \frac{C}{2}(\dot{X}_2-\dot{X}_3)^2 + \frac{C}{2}\dot{X}_3^2$$

$$= \frac{E_j\tau^2}{2}\left[(\dot{x}_2-\dot{x}_3)^2+\dot{x}_3^2\right] \tag{24}$$

In the Hamiltonian approach, unless the kinetic energy is explicitly given in term of canonical momentum, we need to solve a set of linear equations derived from (2) in order to get the canonical form. According to (2) we get,

$$\frac{\partial L}{\partial \dot{x}_2} = \tau^2 E_j(\dot{x}_2-\dot{x}_3) = p_2 \tag{25}$$

$$\frac{\partial L}{\partial \dot{x}_3} = \tau^2 E_j(2\dot{x}_3-\dot{x}_2) = p_3 \tag{26}$$

Solving the above simultaneous equations for $\dot{x}_2$ and $\dot{x}_3$ to be in term of p_2 and p_3, and substitute into (24) we obtain,

$$K_p = \frac{p_2^2+(p_2+p_3)^2}{2E_j\tau^2} \tag{27}$$

According to (4) and (5) the equations of motion are obtainable as follow,

$$\frac{dx_2}{dt} = -\frac{2p_2+p_3}{2E_j\tau^2} \tag{28}$$

$$\frac{dx_3}{dt} = -\frac{p_3}{2E_j\tau^2} \tag{29}$$

$$\frac{dp_2}{dt} = E_j\left(\frac{x_2 - x_1}{A} - \sin(x_2 - x_3)\right) \tag{30}$$

$$\frac{dp_3}{dt} = E_j(\sin(x_3 - x_2) + \sin x_3) \tag{31}$$

This example shows that the equations of motion in the Hamiltonian approach is a set of simultaneous first order differential equations, which are readily solved by numerical method, such as the Runge-Kutta method.

In the following we would like to consider a circuit which illustrate the possible of implicit extraneous variables being introduced into a system, and it can be easily shown that if Lagrangian formalism is adopted for writing the equations of motion, then the equations of motion of this circuit can not retain the same form as the previous example thus posing problem in writing standardized algorithms for the system, however we shall show in the following that for the Hamiltonian formalism, standard form of equations of motion is retained.

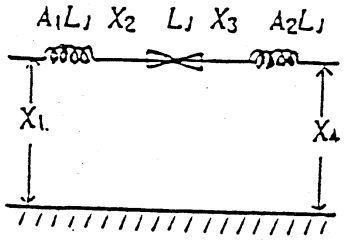


Fig.3

The kinetic energy and the potential energy of the system show in Fig. 3 can be written as follow,

$$U = E_j\left(\frac{(x_1 - x_2)^2}{2A_1} + \frac{(x_3 - x_4)^2}{2A_2} - \cos(x_2 - x_3)\right) \tag{32}$$

$$K = \frac{E_j\tau^2}{2}(\dot{x}_2 - \dot{x}_3)^2 \tag{33}$$

In the Hamiltonian approach, the canonical variables can be obtained by (2) as follow,

$$p_2 = \frac{\partial L}{\partial \dot{x}_2} = E_j\tau^2(\dot{x}_2 - \dot{x}_3) \tag{34}$$

$$p_3 = \frac{\partial L}{\partial \dot{x}_3} = E_j \tau^2 (\dot{x}_3 - \dot{x}_2) \tag{35}$$

eliminating either one of the variables of canonical momentum (e.g. p_3), we obtain the kinetic energy in term of only one canonical momentum as follow,

$$K_p = \frac{p_2^2}{2E_j\tau^2} \tag{36}$$

and the Hamiltonian equation of motion is derivable by (4) and (5) as follow,

$$\frac{dp_2}{dt} = E_j\left(\frac{x_2 - x_1}{A_1} + \sin(x_2 - x_3)\right) \tag{37}$$

$$\frac{dp_3}{dt} = E_j\left(\frac{x_3 - x_4}{A_2} + \sin(x_3 - x_2)\right) \tag{38}$$

$$\frac{dx_2}{dt} = \frac{dx_3}{dt} = \frac{p_2}{E_j\tau^2} \tag{39}$$

The equations of motion still retain the standard form, and consistent computer algebra algorithms can be applied to this problem in the same way as the first example. For this reason and the reason that first order differential equations are indigenous to the Hamiltonian formalism, and are readily be solved by Runge-Kutta method, therefore Hamiltonian formulation is adopted to develop algorithms for automatic circuitry code generator.

6. Design of a Circuit Simulation Code Generator Based on Computer Algebra

In section 4, we have shown the basic principles underlying our approach, however to develop a sophisticated generator which is user friendly we need something additional. Fig.4 gives a slightly more complicated circuit of five DCFPs connected via some delay line. This is a majority logic circuit, which operates on the principle that the output logic state will be decided by the majority input states, for example, if inputs S1 is low and S2 and S3 are high then the logic output at DCFP4 should be high. In order to simulate the circuit

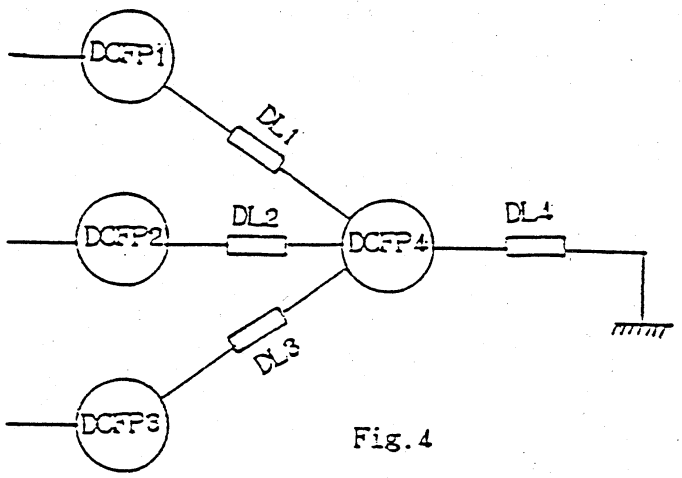


Fig.4

behavior, the user of the generator has only to input the specifications of the circuit instead of writing a REDUCE(3.0) program which is presumably more complicated than the example given in List 1. In short, he only has to write essentially the following:

```
DCFP(1,CK1);
DL(1,X1,X4);
DCFP(2,CK1);
DL(2,X2,X4);
DCFP(3,CK1);
DL(3,X3,X4);
DCFP(4,CK2);
DL(4,X4,0);
```

DCFP(p , CK_n) is the specifications of DCFP; and the first argument is to designate the p-th DCFP number and the second argument is to specify the phase of clock being used to drive the DCFP. DL(i, X_j, X_k) is the specifications of the delay line, and the first argument is an integer designating the i-th delay line, the second and the third arguments are interfacing flux variables at the two ends of the delay line. Essentially, the above specifications will be sufficient to generate FORTRAN code to simulate the circuit operation

of Fig. 4. Looking at the above specifications it is clear that the specifications is simple and in addition , the specifications provides a good correspondence to the graphical drawing of the circuit configuration. Therefore the specifications itself not only serves as specifications but also a good documentation for the circuit diagram.

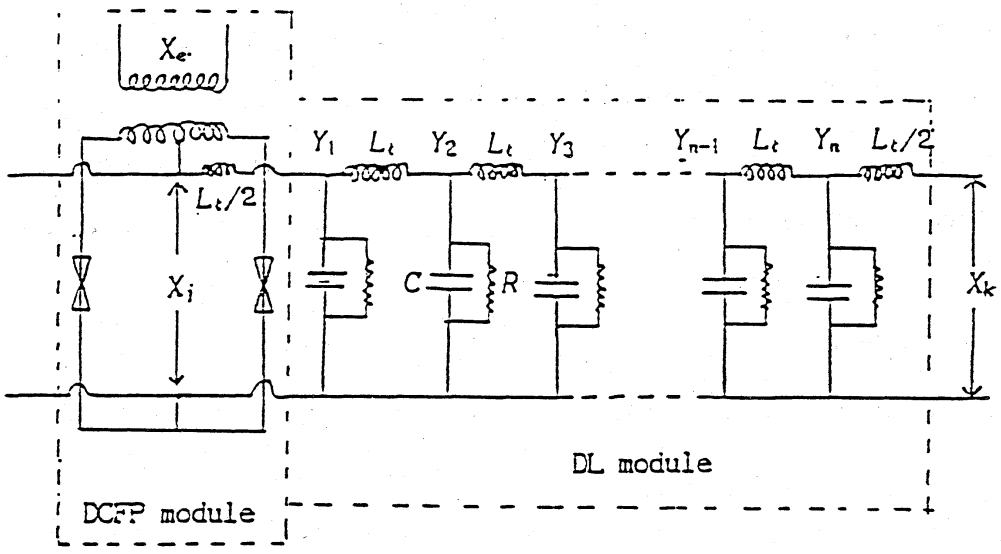


Fig.5

The DCFP(p , CK_n) and DL(i, X_j, X_k) are nothing but sub-Hamiltonians of DCFPs (without leakage inductance) and delay lines respectively and can be easily written down by referring to Fig. 5 as follow,

$$U_{DCFP} = \frac{\Phi_0 I_m}{2\pi}\cos x_e \cos x_j \tag{40}$$

$$U_{DL} = \frac{(X_j - Y_1)^2}{Lt} + \sum_{i=0}^{n-1}\frac{(Y_i - Y_{i+1})^2}{2Lt} + \frac{(Y_n - X_k)^2}{2Lt} \tag{41}$$

$$K_{DCFP} = \frac{C}{2}(\dot{X}_j - \dot{X}_e)^2 + \frac{C}{2}(\dot{X}_j + \dot{X}_e)^2 \tag{42}$$

$$K_{DL} = \sum_{i=1}^{m}\frac{C}{2}\dot{Y}_i^2 \tag{43}$$

$$D_{DCFP} = \frac{1}{2R}(\dot{X}_j - \dot{X}_e)^2 + \frac{1}{2R}(\dot{X}_j + \dot{X}_e) \tag{44}$$

$$D_{DL} = \sum_{i=1}^{m}\frac{1}{2Rt}\dot{Y}_i^2 \tag{45}$$

$$H_{DCFP} = K_{DCFP} + U_{DCFP} \tag{46}$$

and

$$H_{DL} = K_{DL} + U_{DL} \tag{47}$$

The above expressions can be converted into computer algebra algorithms writing in REDUCE(3.0) statements. The number of code lines of FORTRAN program generated from the above specifications are presumably many times the specification statements. Therefore a user who needs only to write the specifications, a considerable saving of time and effort are obvious. If we are to write FORTRAN code for every circuit configuration to be simulated, then let alone the enormous work for coding and the various changes we have to make for every circuit configurations , the chances to make error will be very high for a complicated circuit configuration.

The system has already been implemented in one of the REDUCE(3.0) system for actual circuit simulation and design. The detail of the implementation and application of the system can be found in reference[3] As we have mentioned earlier that system is extensible, if a user intends to incorporate a new logic function into the CSCG then he has only to write a similar sub-Hamiltonian of his own and coded in REDUCE(3.0) statements as a procedure and added to the CSCG.

7. Conclusion

From what have been discussed we conclude that the simplicity and extensibility of the system and the ability of the system to handle complicated circuit dynamics are the direct consequences of the Hamiltonian formalism, which enable the total system to be partitioned into sub-Hamiltonian, and equally importance is the power of computer algebra system such as REDUCE(3.0). The generator can also be extended to include mechanics

systems since in many mechanics systems it is possible to specify systems by Hamiltonian.

References

1. K. F. Loe and E. Goto, *Analysis of Flux Input Output Josephson Pair Device,* RIKEN Symposium on Josephson Junction Electronic, March 1984.

2. Anthony C. Hearn, *REDUCE User's Manual version 3.0,* The Rand Corporation, Santa Monica, CA., April 1983.

3. N. Ohsawa, K. F. Loe, and E. Goto, *Implementation and Applications of Circuit Simulation Code Generator,* RIKEN (IPCR) Information Science Lab., August 1984. Preprint

List 1

```
 1 ;
 2 %
 3 %  INPUT
 4 %
 5 ;
 6
 7  K   := 1/(2*M)*P**2;
 8  U   := KO/2*Q**2;
 9  D   := B/2*QDOT**2;
10  H   := K + U;
11
12 ;
13 %
14 %   RUNGE-KUTTA METHOD
15 %
16 ;
17
18  PROCEDURE RUNGEKUTTA(F1, F2, P,  Q, TT);
19   BEGIN
20    SCALAR K11, K12, K21, K22, K31, K32, K41, K42;
21
22       K11 := HH*F1;
23       K12 := HH*F2;
24       K21 := HH*SUB(TT=TT+HH/2, P=P+K11/2, Q=Q+K12/2, F1);
25       K22 := HH*SUB(TT=TT+HH/2, P=P+K11/2, Q=Q+K12/2, F2);
26       K31 := HH*SUB(TT=TT+HH/2, P=P+K21/2, Q=Q+K22/2, F1);
27       K32 := HH*SUB(TT=TT+HH/2, P=P+K21/2, Q=Q+K22/2, F2);
28       K41 := HH*SUB(TT=TT+HH  , P=P+K31  , Q=Q+K32  , F1);
29       K42 := HH*SUB(TT=TT+HH  , P=P+K31  , Q=Q+K32  , F2);
30       PN  := P + (K11 + 2*K21 + 2*K31 + K41)/6;
31       QN  := Q + (K12 + 2*K22 + 2*K32 + K42)/6;
32   END;
33
34 ;
35 %
36 %  HAMILTONIAN CALCULATION
37 %
38 ;
39
40  DIFP := -DF(H,Q)-SUB(QDOT=P/M,DF(D,QDOT));
41  DIFQ := DF(H,P);
42
43  RUNGEKUTTA(DIFP, DIFQ, P, Q, TT);
44
45 ;
46 %
47 %
48 % FORTRAN PROGRAM OUTPUT
49 %
50 %
51 ;
52  OFF ECHO*
53  ON FORT*
54  OUT OUTFILE;
55
56  WRITE "      PROGRAM RUNGE";
57  WRITE "*";
58  WRITE "* INPUT";
59  WRITE "*";
60  WRITE "      IMPLICIT REAL(K,M)";
61  WRITE "      WRITE(6,*) ' INITIAL VALUE OF P'";
62  WRITE "      READ(5,*) P";
63  WRITE "      WRITE(6,*) ' P = ',P";
64  WRITE "      WRITE(6,*) ' INITIAL VALUE OF Q'";
65  WRITE "      READ(5,*) Q";
66  WRITE "      WRITE(6,*) ' Q = ',Q";
67  WRITE "      WRITE(6,*) ' VALUE OF M'";
68  WRITE "      READ(5,*) M";
69  WRITE "      WRITE(6,*) ' M = ',M";
70  WRITE "      WRITE(6,*) ' VALUE OF KO'";
71  WRITE "      READ(5,*) KO";
72  WRITE "      WRITE(6,*) ' KO = ',KO";
73  WRITE "      WRITE(6,*) ' VALUE OF B'";
74  WRITE "      READ(5,*) B";
75  WRITE "      WRITE(6,*) ' B = ',B";
76  WRITE "      WRITE(6,*) ' STEP SIZE OF T'";
77  WRITE "      READ(5,*) HH";
78  WRITE "      WRITE(6,*) ' STEP SIZE OF T = ',HH";
79  WRITE "      WRITE(6,*) ' FINAL VALUE OF T ?'";
80  WRITE "      READ(5,*) TF";
81  WRITE "      WRITE(6,*) ' FINAL VALUE OF T = ',TF";
```

List 1

```
 82  WRITE "*";
 83  WRITE "*  INITIALIZATION";
 84  WRITE "*";
 85  WRITE "       TT = 0";
 86  WRITE "       WRITE(9,*) ' H = ",H,"'";
 87  WRITE "       WRITE(9,*) ' D = ",D,"'";
 88  WRITE "       WRITE(9,901) M,KO,B";
 89  WRITE "  901 FORMAT(' M  = ',E20.10/' KO = ',E20.10/' B  = ',E20.10)";
 90  WRITE "       WRITE(9,910) TT,Q,P";
 91  WRITE "  910 FORMAT(' ',3E20.10)";
 92  WRITE "*";
 93  WRITE "*  LOOP";
 94  WRITE "*";
 95  WRITE "  100 CONTINUE";
 96  WRITE "           PN=",PN;
 97  WRITE "           Q=",QN;
 98  WRITE "           P = PN";
 99  WRITE "           TT = TT + HH";
100  WRITE "           WRITE(9,910) TT,Q,P";
101  WRITE "       IF ( TT .LT. TF ) GO TO 100";
102  WRITE "*";
103  WRITE "       STOP";
104  WRITE "       END";
105  SHUT;
106  OFF FORT;
107  END;
```

List 2

```
 1        PROGRAM RUNGE
 2 *
 3 * INPUT
 4 *
 5        IMPLICIT REAL(K,M)
 6        WRITE(6,*) ' INITIAL VALUE OF P'
 7        READ(5,*) P
 8        WRITE(6,*) ' P = ',P
 9        WRITE(6,*) ' INITIAL VALUE OF Q'
10        READ(5,*) Q
11        WRITE(6,*) ' Q = ',Q
12        WRITE(6,*) ' VALUE OF M'
13        READ(5,*) M
14        WRITE(6,*) ' M = ',M
15        WRITE(6,*) ' VALUE OF KO'
16        READ(5,*) KO
17        WRITE(6,*) ' KO = ',KO
18        WRITE(6,*) ' VALUE OF B'
19        READ(5,*) B
20        WRITE(6,*) ' B = ',B
21        WRITE(6,*) ' STEP SIZE OF T'
22        READ(5,*) HH
23        WRITE(6,*) ' STEP SIZE OF T = ',HH
24        WRITE(6,*) ' FINAL VALUE OF T ?'
25        READ(5,*) TF
26        WRITE(6,*) ' FINAL VALUE OF T = ',TF
27 *
28 *  INITIALIZATION
29 *
30        TT = 0
31        WRITE(9,*) ' H = (KO*M*Q**2+P**2)/(2.*M)'
32        WRITE(9,*) ' D = (B*QDOT**2)/2.'
33        WRITE(9,901) M,KO,B
34   901  FORMAT(' M  = ',E20.10/' KO = ',E20.10/' B  = ',E20.10)
35        WRITE(9,910) TT,Q,P
36   910  FORMAT(' ',3E20.10)
37 *
38 *  LOOP
39 *
40   100  CONTINUE
41             PN=(B**4*HH**4*P+B**3*HH**4*KO*M*Q-4.*B**3*HH**3*M*P-
42       . 3.*B**2*HH**4*KO*M*P-4.*B**2*HH**3*KO*M**2*Q+12.*B**2*HH**
43       . 2*M**2*P-2.*B*HH**4*KO**2*M**2*Q+8.*B*HH**3*KO*M**2*P+12.
44       . *B*HH**2*KO*M**3*Q-24.*B*HH*M**3*P+HH**4*KO**2*M**2*P+4.*
45       . HH**3*KO**2*M**3*Q-12.*HH**2*KO*M**3*P-24.*HH*KO*M**4*Q+
46       . 24.*M**4*P)/(24.*M**4)
47             Q=(-B**3*HH**4*P-B**2*HH**4*KO*M*Q+4.*B**2*HH**3*M*P+
48       . 2.*B*HH**4*KO*M*P+4.*B*HH**3*KO*M**2*Q-12.*B*HH**2*M**2*P+
49       . HH**4*KO**2*M**2*Q-4.*HH**3*KO*M**2*P-12.*HH**2*KO*M**3*Q
50       . +24.*HH*M**3*P+24.*M**4*Q)/(24.*M**4)
51             P = PN
52             TT = TT + HH
53             WRITE(9,910) TT,Q,P
54        IF ( TT .LT. TF ) GO TO 100
55 *
56        STOP
57        END
```

H = (KO*M*Q**2+P**2)/(2.*M)

D = (B*QDOT**2)/2.

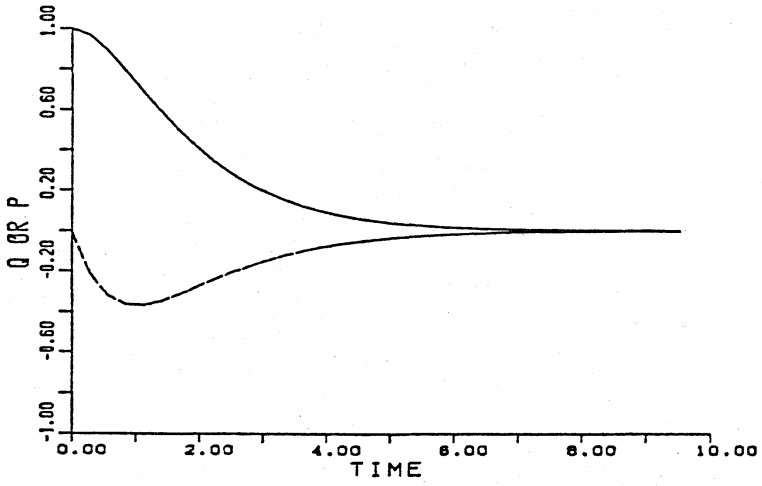
M = 0.100000000 E+01

KO = 0.100000000 E+01

B = 0.200000000 E+01

——————————————— : P

——————————————— : Q



H = (KO*M*Q**2+P**2)/(2.*M)

D = (B*QDOT**2)/2.

M = 0.100000000 E+01

KO = 0.100000000 E+01

B = 0.500000000 E+00

——————————————— : P

——————————————— : Q